Docket No.: 67161-118 **PATENT**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of	:	Customer Number: 20277
Isao MINEMATSU, et al.	:	Confirmation Number:
Serial No.:	:	Group Art Unit:
Filed: October 17, 2003	:	Examiner: Unknown

For: SEMICONDUCTOR MEMORY DEVICE STORING PART OF PROGRAM DESIGNATED BY PROGRAMMER, AND SOFTWARE DEVELOPMENT APPARATUS FOR SYSTEM USING THE SAME

CLAIM OF PRIORITY AND
TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENT

Mail Stop CPD
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

In accordance with the provisions of 35 U.S.C. 119, Applicants hereby claim the priority of:

Japanese Patent Application No. 2003-002233, filed January 8, 2003

cited in the Declaration of the present application. A certified copy is submitted herewith.

Respectfully submitted,

MCDERMOTT, WILL & EMERY

Stephen A. Becker
Registration No. 26,527

600 13th Street, N.W.
Washington, DC 20005-3096
(202) 756-8000 SAB:tlb
Facsimile: (202) 756-8087
Date: October 17, 2003

日　本　国　特　許　庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２００３年　１月　８日

出願番号
Application Number: 特願２００３－００２２３３

[ST.10/C]: [ＪＰ２００３－００２２３３]

出願人
Applicant(s): 三菱電機株式会社

２００３年　２月　７日

特許庁長官
Commissioner,
Japan Patent Office 太田信一郎

出証番号　出証特２００３－３００５６４９

【書類名】　特許願
【整理番号】　541745JP01
【提出日】　平成15年 1月 8日
【あて先】　特許庁長官殿
【国際特許分類】　G06F 12/08
【発明者】
【住所又は居所】　東京都千代田区丸の内二丁目２番３号　三菱電機株式会社内
【氏名】　峯松　勲
【発明者】
【住所又は居所】　東京都千代田区丸の内二丁目２番３号　三菱電機株式会社内
【氏名】　佐藤　尚和
【特許出願人】
【識別番号】　000006013
【氏名又は名称】　三菱電機株式会社
【代理人】
【識別番号】　100064746
【弁理士】
【氏名又は名称】　深見　久郎
【選任した代理人】
【識別番号】　100085132
【弁理士】
【氏名又は名称】　森田　俊雄
【選任した代理人】
【識別番号】　100083703
【弁理士】
【氏名又は名称】　仲村　義平

【選任した代理人】

【識別番号】　100096781

【弁理士】

【氏名又は名称】　堀井　豊

【選任した代理人】

【識別番号】　100098316

【弁理士】

【氏名又は名称】　野田　久登

【選任した代理人】

【識別番号】　100109162

【弁理士】

【氏名又は名称】　酒井　將行

【手数料の表示】

【予納台帳番号】　008693

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　半導体記憶装置およびソフトウェア開発装置

【特許請求の範囲】

【請求項１】　プロセッサと命令キャッシュとの間に接続される半導体記憶装置であって、

プロセッサによって実行される命令列の一部が格納される命令バッファと、

前記命令バッファに格納された命令列のアドレス範囲が設定されるアドレステーブルと、

前記プロセッサから出力された命令アドレスが前記アドレステーブルに設定されたアドレス範囲内であるか否かを判定する判定器と、

前記判定器の判定結果に応じて、前記命令バッファに格納される命令コードと前記命令キャッシュに格納される命令コードとを選択的に出力するセレクタとを含む、半導体記憶装置。

【請求項２】　前記判定器は、非動作モードが設定されている場合には、前記プロセッサと前記命令キャッシュとが直接接続されるように制御を行なう、請求項１記載の半導体記憶装置。

【請求項３】　非動作モードが設定されている場合には、前記アドレステーブルおよび前記命令バッファが前記プロセッサのメモリマップ上にマッピングされ、メモリマップドデバイスとして動作する、請求項１記載の半導体記憶装置。

【請求項４】　前記判定器は、前記プロセッサが出力した命令アドレスから、前記アドレステーブルに設定されたアドレスの最大値を減算する第１の減算器と、

前記プロセッサが出力した命令アドレスから、前記アドレステーブルに設定されたアドレスの最小値を減算する第２の減算器と、

前記第１の減算器および前記第２の減算器による減算結果の符号に応じて、前記プロセッサから出力された命令アドレスが前記アドレステーブルに設定されたアドレス範囲内であるか否かを判定する論理回路とを含む、請求項１記載の半導体記憶装置。

【請求項５】　前記第２の減算器は、減算結果をアドレスとして前記命令バ

ッファへ出力する、請求項４記載の半導体記憶装置。

【請求項６】　前記判定器はさらに、前記プロセッサから出力された命令アドレスの上位ビットと、前記アドレステーブルに設定された最大値の上位ビットとを比較し、一致しない場合には前記第１の減算器および前記第２の減算器の動作を停止させる比較器を含む、請求項４記載の半導体記憶装置。

【請求項７】　前記論理回路は、前記プロセッサから出力された命令アドレスが前記アドレステーブルに設定されたアドレス範囲内であると判定した場合には、前記プロセッサから出力された命令アドレスが前記アドレステーブルに設定されたアドレス範囲外であると判定するまで前記第２の減算器の動作を停止させる、請求項４記載の半導体記憶装置。

【請求項８】　前記プロセッサから出力される命令アドレスが前記命令バッファのアドレスとして供給される、請求項７記載の半導体記憶装置。

【請求項９】　プロセッサと命令キャッシュとの間に接続され、前記プロセッサによって実行される命令列の一部を格納する半導体記憶装置を含んだプロセッサシステムのソフトウェア開発装置であって、

ソースファイルの中から、前記半導体記憶装置に配置する命令列を抽出するための抽出手段と、

前記抽出手段によって抽出された命令列が本来マップされるアドレス情報を取得するための取得手段と、

前記取得手段によって取得されたアドレス情報を前記ソースファイルと互換性のある形式に整形するための整形手段と、

前記ソースファイル、前記抽出手段によって抽出された命令列および前記整形手段によって整形されたアドレス情報からロードモジュールを生成するための生成手段とを含む、ソフトウェア開発装置。

【請求項１０】　前記抽出手段は、前記ソースファイルの中から所定の予約語で指定された命令列を抽出して、前記半導体記憶装置に配置する命令列とする、請求項９記載のソフトウェア開発装置。

【請求項１１】　前記抽出手段は、前記ソースファイルの中から共通命令列を抽出して、前記半導体記憶装置に配置する命令列とする、請求項９記載のソフ

トウェア開発装置。

【請求項１２】　前記取得手段は、取得したアドレス情報から命令列の先頭アドレスの下位ビットを抽出し、当該先頭アドレスの下位ビットに相当するバイト数だけ命令列のアドレスをずらす、請求項９～１１のいずれかに記載のソフトウェア開発装置。

【請求項１３】　前記生成手段は、前記抽出手段によって抽出された命令列に対応する主記憶の領域の先頭アドレスに、前記主記憶に予め設けられた予約領域の先頭へ分岐する分岐命令を設定し、

前記抽出手段によって抽出された命令列の最後尾に、当該命令列に対応する主記憶の領域の最後尾アドレスの直後の命令へ分岐する分岐命令を追加する、請求項９～１１のいずれかに記載のソフトウェア開発装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、ＣＰＵ（Central Processing Unit）と共に使用される半導体記憶装置に関し、特に、プログラマによって指定されたプログラムの一部を格納し、ＣＰＵの命令フェッチ時に命令コードを出力する半導体記憶装置およびそれを用いたシステムのソフトウェア開発装置に関する。

【０００２】

【従来の技術】

近年、ＣＰＵの高速化が進み、キャッシュメモリを搭載したＣＰＵが盛んに開発されている。一般に、キャッシュメモリは、ＣＰＵと主記憶との間に接続され、ＣＰＵが最近利用した主記憶の内容を一時的に保持する小規模、高速の記憶装置である。キャッシュは、保持する内容がＣＰＵの命令である場合には“命令キャッシュ”、データである場合には“データキャッシュ”、命令およびデータの両方である場合には“ユニファイドキャッシュ”というように、３種類に分類される。

【０００３】

キャッシュは、ＣＰＵによるメモリへのアクセスが局所的であること活用した

仕組みであり、概してＣＰＵの処理性能を向上させる効果が得られることが広く知られている。ただし、その効果はＣＰＵが実行するプログラムに依存する。たとえば、ＣＰＵが実行するプログラムの処理の特徴として、メモリアクセスに規則性がない場合には、キャッシュのヒット率が低くなって性能の向上を図ることができない。

【０００４】

また、キャッシュは、ＣＰＵから出力された参照アドレスとキャッシュの内容との一致検出を行なう仕組みを本質的に持っているため、消費電力が大きくなる傾向にある。そのため、キャッシュを導入しても、電力効率（処理性能／消費電力）の面で必ずしも向上するとは限らない。

【０００５】

さらには、キャッシュの内容はＣＰＵによるプログラムの実行経歴に依存するため、同じアドレスにアクセスする場合でも、キャッシュにヒットする場合とミスヒットする場合とがあり、命令やデータのアクセスサイクルが保証されない。このため、キャッシュを用いたシステムにおいて、リアルタイム性の高いソフトウェアの最適化が困難である。

【０００６】

これらの問題点を解決するための技術として、以下のようなものが開示されている。

【０００７】

【特許文献１】

特開平１０－３４０２２６号公報

【０００８】

【特許文献２】

特開平９－３１９６５７号公報

【０００９】

【特許文献３】

ＵＳＰ５，３８１，５３３

【００１０】

【非特許文献１】

P.R.Panda et al., "Efficient Utilization of Scratch-Pad Memory in Embedded Processor Applications", European Design and Test Conference, March 1997.

【００１１】

【発明が解決しようとする課題】

特許文献１は、タグメモリが、アドレスタグ中の各ウェイを通じて共通なビット群を有する第１のタグメモリと、各ウェイ毎に独自なビット群を有する第２のタグメモリとに分割され、データ処理装置から示されるアドレスを分割して第１および第２のタグメモリ毎に比較することにより、ヒット率を低下させることなくマイクロプロセッサの消費電力を削減するものである。しかし、電力効率の向上に寄与する反面、プログラムの実行時における性能向上には寄与しないといった問題点があった。

【００１２】

また、特許文献２においては、命令キャッシュが１ライン１命令で構成され、命令キャッシュと主記憶との間に命令ストリームバッファが設けられる。命令ストリームバッファは、キャッシュラインの整数倍の大きさで、主記憶から読込んだ可変長の命令列の書込みが可能であり、命令キャッシュへの出力はキャッシュライン単位で可能なように構成するものである。これによって、無駄な命令の読込みをなくすことができるが、電力効率の向上を図ることができず、命令のアクセスサイクルが保証されないといった問題点があった。

【００１３】

また、特許文献３においては、それぞれの命令トレースセグメントが命令のブロックを含み、それぞれのブロックの最初の命令が分岐命令の次の命令となり、それぞれのブロックの最後の命令が分岐命令となるように配置される。これによって、ヒット率を高め、電力効率を向上させるものである。しかし、特許文献２と同様に、命令やデータのアクセスサイクルが保証されない。

【００１４】

さらには、非特許文献１においては、主記憶と異なるアドレス空間に、小規模

、高速のスクラッチパッドメモリ（以下、ＳＰＭと略す。）が配置される。そして、プログラマが、頻繁にアクセスされる命令やデータを指定して、ＳＰＭに配置するものである。これによって、ＳＰＭに対するアクセスサイクル数が保証され、キャッシュに含まれる一致検出機能が不要となるため消費電力を少なくすることができる。しかし、一般に、ＳＰＭはデータメモリ用、すなわちデータキャッシュの代用とされ、命令メモリ用として実装することが困難であるといった問題点があった。

【００１５】

本発明は、上記問題点を解決するためになされたものであり、その目的は、命令フェッチ時におけるアクセスサイクルを保証でき、プロセッサシステムの電力効率を向上させることが可能な半導体記憶装置およびその半導体記憶装置を含んだプロセッサシステムのソフトウェア開発を効率的に行なえるソフトウェア開発装置を提供することである。

【００１６】

【課題を解決するための手段】

本発明のある局面に従えば、プロセッサと命令キャッシュとの間に接続される半導体記憶装置であって、プロセッサによって実行される命令列の一部が格納される命令バッファと、命令バッファに格納された命令列のアドレス範囲が設定されるアドレステーブルと、プロセッサから出力された命令アドレスがアドレステーブルに設定されたアドレス範囲内であるか否かを判定する判定器と、判定器の判定結果に応じて、命令バッファに格納される命令コードと命令キャッシュに格納される命令コードとを選択的に出力するセレクタとを含む。

【００１７】

本発明の別の局面に従えば、プロセッサと命令キャッシュとの間に接続され、プロセッサによって実行される命令列の一部を格納する半導体記憶装置を含んだプロセッサシステムのソフトウェア開発装置であって、ソースファイルの中から、半導体記憶装置に配置する命令列を抽出するための抽出手段と、抽出手段によって抽出された命令列が本来マップされるアドレス情報を取得するための取得手段と、取得手段によって取得されたアドレス情報をソースファイルと互換性のあ

る形式に整形するための整形手段と、ソースファイル、抽出手段によって抽出された命令列および整形手段によって整形されたアドレス情報からロードモジュールを生成するための生成手段とを含む。

【００１８】

【発明の実施の形態】

（第１の実施の形態）

図１は、本発明の第１の実施の形態における半導体記憶装置が利用されるプロセッサシステムの構成例を示すブロック図である。このプロセッサシステムは、半導体記憶装置（図面においてはＳＴＡＳＨと表記する。）１と、ＣＰＵコア２と、命令キャッシュ３と、データキャッシュ４と、主記憶５と、ＵＳＡＲＴ（Universal Synchronous and Asynchronous Receiver-Transmitter）６とを含む。

【００１９】

半導体記憶装置１は、ＣＰＵコア２と命令キャッシュ３との間に接続されると共に、システムバス７にも接続される。ＣＰＵコア２は、半導体記憶装置１から直接命令をフェッチできると共に、半導体記憶装置１を介して命令キャッシュ３から命令をフェッチすることができる。また、ＣＰＵコア２は、データキャッシュ４に対してデータの読出し、書込みが可能である。なお、ＣＰＵコア２は、システムバス７を介して半導体記憶装置１にアクセスすることも可能である。

【００２０】

命令キャッシュ３およびデータキャッシュ４は、ブリッジ８を介してシステムバス７に接続される。また、システムバス７には、主記憶５およびＵＳＡＲＴ６などの各種の機能ブロックが接続される。図１においては、各種の機能ブロックを代表してＵＳＡＲＴ６が記載されている。なお、図１においては、全ての機能ブロックが単一のＬＳＩ（Large Scale Integrated circuit）チップによって実現されているが、一部の機能が別のＬＳＩチップによって実現されてもよいし、半導体記憶装置１がブリッジ８に直接接続される構成であってもよい。

【００２１】

図２は、本発明の第１の実施の形態における半導体記憶装置１の内部構成を示すブロック図である。この半導体記憶装置１は、アドレステーブル１１と、判定

器１２と、命令バッファ１３と、命令バッファ１３から出力される命令コードおよび命令キャッシュ３から出力される命令コードを選択的に出力するセレクタ１４と、スイッチ１（以下、ＳＷ１と略す。）１５およびスイッチ２（以下、ＳＷ２と略す。）１６と、インバータ１７とを含む。

【００２２】

命令バッファ１３は、ランダムアクセスが可能なメモリデバイスであり、ＣＰＵコア２が実行する命令列の一部を格納する。

【００２３】

アドレステーブル１１は、有効ビット１１１と、アドレスの最大値１１２と、アドレスの最小値１１３とが格納される領域を含む。アドレスの最大値１１２には、命令バッファ１３に格納された命令列に対応するアドレスの最大値が格納される。また、アドレスの最小値１１３には、命令バッファ１３に格納された命令列に対応するアドレスの最小値が格納される。有効ビット１１１には、アドレステーブルに格納されたアドレス情報が有効であるか、無効であるかを示すビットが格納される。

【００２４】

判定器１２は、アドレステーブル１１に格納されたアドレスと、ＣＰＵコア２から出力されたアドレスとを参照して、命令バッファ１３から出力された命令コードを選択するか、命令キャッシュ３から出力された命令コードを選択するかを判定する。この判定器１２は、減算器１２１および１２２と、減算器１２１による減算結果が格納される差レジスタ１（１２３）と、減算器１２２による減算結果が格納される差レジスタ２（１２４）と、インバータ１２５と、３入力ＡＮＤ回路１２６とを含む。

【００２５】

減算器１２１は、ＣＰＵコア２が出力したアドレスからアドレスの最大値１１２を減算し、その減算結果を差レジスタ１（１２３）に格納する。また、減算器１２２は、ＣＰＵコア２が出力したアドレスからアドレスの最小値１１３を減算し、その減算結果を差レジスタ２（１２４）に格納する。

【００２６】

ＡＮＤ回路１２６は、有効ビット１１１の値と、差レジスタ１（１２３）の符号ビット（ＭＳＢ：Most Significant Bit）と、差レジスタ２（１２４）の符号ビットを反転した値との論理和を演算することにより、ＣＰＵコア２が出力したアドレスが所定の範囲に含まれているか否かを判定する。すなわち、“最小値１１３≦命令アドレス”、かつ、“命令アドレス＜最大値１１２”、かつ、“有効ビット＝１”の場合には命令アドレスがヒットしたとして、ＡＮＤ回路１２６はハイレベル（以下、Ｈレベルと略す。）を出力する。

【００２７】

命令アドレスがヒットしたと判定された場合には、ＡＮＤ回路１２６がＨレベルを出力することにより、セレクタ１４に命令バッファ１３から出力される命令コードを選択させ、ＳＷ２（１６）をオンして差レジスタ２（１２４）からの値（命令バッファ１３のアドレス）を命令バッファ１３へ出力させると共に、命令キャッシュ３へのアドレス出力が不要であるのでＳＷ１（１５）をオフする。

【００２８】

また、命令アドレスがミスヒットと判定された場合には、ＡＮＤ回路１２６がロウレベル（以下、Ｌレベルと略す。）を出力することにより、セレクタ１４に命令キャッシュ３からの命令コードを選択させ、命令バッファ１３へのアドレス出力が不要であるのでＳＷ２（１６）をオフすると共に、ＳＷ１（１５）をオンして命令アドレスを命令キャッシュ３へ出力させる。

【００２９】

図３は、ＣＰＵコア２が半導体記憶装置１をメモリマップドデバイスとしてアクセスする場合のメモリマップを示す図である。制御レジスタ“ＳＴＡＳＨ＿ＳＴＳ”は、半導体記憶装置１の動作状況（アクティブ／非アクティブ）を示すレジスタである。また、制御レジスタ“ＳＴＡＳＨ＿ＣＮＴ”は、半導体記憶装置１の動作状況を設定するためのレジスタである。

【００３０】

図３に示すメモリマップのうち、“アドレステーブル”および“命令バッファ”として示されるアドレス空間は、図２の命令バッファ１３およびアドレステーブル１１の写像であり、本実施の形態においては各々４Ｋバイトの領域となって

いる。

【００３１】

これらのアドレス空間は、半導体記憶装置１が非アクティブの状態でのみ読出し／書込みが可能である。制御レジスタ“ＳＴＡＳＨ＿ＣＮＴ”に値を設定することにより、これらのアドレス空間を非アクティブ状態にしたり、アクティブ状態にしたりすることが可能である。半導体記憶装置１が非アクティブ状態の場合、判定器１２は動作せず、半導体記憶装置１がバイパスされて、ＣＰＵコア２と命令キャッシュ３とが直接接続される。

【００３２】

アドレステーブルの０ｘ４０００＿１０００番地に最大値１１２が設定され、０ｘ４０００＿１００４番地に最小値１１３が設定される。半導体記憶装置１のリセット直後において、アドレステーブル１１は“０”に初期化される。アドレステーブル１１に“０”以外の値が設定されたときに、アドレステーブル１１の有効ビット１１１が自動的に設定される。

【００３３】

図４は、本発明の第１の実施の形態における半導体記憶装置１を用いたプロセッサシステムのソフトウェア開発装置の構成を示すブロック図である。このソフトウェア開発装置は、コンピュータ本体２１、ディスプレイ装置２２、ＦＤ（Flexible Disk）２４が装着されるＦＤドライブ２３、キーボード２５、マウス２６、ＣＤ－ＲＯＭ（Compact Disc-Read Only Memory）２８が装着されるＣＤ－ＲＯＭ装置２７およびネットワーク通信装置２９を含む。

【００３４】

ソフトウェア開発装置を実現するプログラム（以下、ソフトウェア開発プログラムと呼ぶ。）は、ＦＤ２４またはＣＤ－ＲＯＭ２８等の記録媒体によって供給される。ソフトウェア開発プログラムがコンピュータ本体２１によって実行されることにより、ソフトウェアの開発が行なわれる。また、ソフトウェア開発プログラムは他のコンピュータよりネットワーク通信装置２９を経由し、コンピュータ本体２１に供給されてもよい。

【００３５】

コンピュータ本体２１は、ＣＰＵ３０、ＲＯＭ（Read Only Memory）３１、ＲＡＭ（Random Access Memory）３２およびハードディスク３３を含む。ＣＰＵ３０は、ディスプレイ装置２２、ＦＤドライブ２３、キーボード２５、マウス２６、ＣＤ－ＲＯＭ装置２７、ネットワーク通信装置２９、ＲＯＭ３１、ＲＡＭ３２またはハードディスク３３との間でデータを入出力しながら処理を行う。ＦＤ２４またはＣＤ－ＲＯＭ２８に記録されたソフトウェア開発プログラムは、ＣＰＵ３０によりＦＤドライブ２３またはＣＤ－ＲＯＭ装置２７を介して一旦ハードディスク３３に格納される。ＣＰＵ３０は、ハードディスク３３から適宜ソフトウェア開発プログラムをＲＡＭ３２にロードして実行することによって、ソフトウェアの開発が行なわれる。

【００３６】

図５は、本発明の第１の実施の形態におけるソフトウェア開発装置の機能的構成を示すブロック図である。このソフトウェア開発装置は、ソースファイル（アセンブラソース）が格納されるソースファイル格納部１０１と、半導体記憶装置１に配置される命令列（プログラム）を抽出する命令列抽出部１０２と、命令列抽出部１０２によって抽出された命令列が本来マップされるアドレス情報を取得するアドレス情報取得部１０３と、アドレス情報取得部１０３によって取得されたアドレス情報を、アセンブラソースと互換性のある形式で整形するアドレス情報整形部１０４と、アセンブラ１０５と、リンカ１０６とを含む。

【００３７】

図６は、本発明の第１の実施の形態におけるソフトウェア開発装置の処理手順を説明するためのフローチャートである。図６においては、アセンブリ言語でのプログラム開発を想定している。まず、プログラマが、キーボード２５、マウス２６などを用いて半導体記憶装置１に配置するべきプログラムの箇所を指定する（Ｓ１１）。

【００３８】

図７（ａ）は、ＣＰＵコア２によって実行されるプログラムの一部を示す図である。半導体記憶装置１に配置される箇所は、関数“＿ｆｕｎｃ１”の中の“＿ｓｔａｓｈ１＿ｔｏｐ”と“＿ｓｔａｓｈ１＿ｅｎｄ”とで囲まれた部分である

。ここで、“＿ｓｔａｓｈ１＿ｔｏｐ”と“＿ｓｔａｓｈ１＿ｅｎｄ”とは、予約されたシンボルを示している。

【００３９】

次に、命令列抽出部１０２は、ソースファイル格納部１０１に格納されたアセンブラソースから、半導体記憶装置１に配置されるプログラムをスクリプトプログラムで抽出する（Ｓ１２）。

【００４０】

図７（ｂ）は、命令列抽出部１０２によって抽出されたプログラムの一例を示す図である。“＿ｓｔａｓｈ１＿ｔｏｐ”と“＿ｓｔａｓｈ１＿ｅｎｄ”との間のプログラムが抽出されている。

【００４１】

次に、アセンブラ１０５は、ソースファイル格納部１０１に格納されるオリジナルのプログラムをアセンブルする。そして、リンカ１０６は、アセンブラ１０５によってアセンブルされたオリジナルのプログラムをリンクする（Ｓ１３）。このアセンブルおよびリンクは、アセンブリ言語を用いた通常のソフトウェア開発と同様であるので、詳細な説明は行なわない。ただし、後述する処理に備えて、再リンク可能なようにリンクが行なわれる。

【００４２】

次に、アドレス情報取得部１０３は、リンカ１０６によってリンクされたオリジナルのプログラムから、半導体記憶装置１に配置される命令列が本来マップされるアドレス情報を取得する（Ｓ１４）。具体的には、リンカ１０６によってリンクされたプログラムをディスアセンブルするか、マップ情報を抽出することで、予約されたシンボル“＿ｓｔａｓｈ１＿ｔｏｐ”および“＿ｓｔａｓｈ１＿ｅｎｄ”のアドレス情報を取得する。

【００４３】

次に、アセンブラ１０５は、命令列抽出部１０２によって抽出された命令列をアセンブルする（Ｓ１５）。このアセンブルは、通常のアセンブリ言語で記述されたプログラムのアセンブルと同様であるので、詳細な説明は行なわない。

【００４４】

次に、アドレス情報整形部１０４は、アドレス情報取得部１０３によって取得されたアドレス情報を、アセンブラソースと互換性のある形式に整形する（Ｓ１６）。図７（ｃ）は、アドレス情報整形部１０４によって整形されたアドレス情報を示す図である。半導体記憶装置１に配置されるプログラムの先頭アドレスと最終アドレスとが２行目に記述されている。

【００４５】

次に、アセンブラ１０５は、アドレス情報整形部１０４によって整形されたアドレス情報をアセンブルする（Ｓ１７）。このアセンブルは、通常のアセンブリ言語で記述されたプログラムのアセンブルと同様であるので、詳細な説明は行なわない。

最後に、リンカ１０６は、ステップＳ１３，Ｓ１５およびＳ１７において生成されたオブジェクトファイルをリンクして、ロードモジュールを生成する（Ｓ１８）。図７（ｄ）は、リンカ１０６によってリンクされたモジュールのディスアセンブル結果の一部を示す図である。半導体記憶装置１に配置される命令列と、それに対応するアドレス情報とが、それぞれ０ｘ４０００２０００番地と、０ｘ４０００１０００番地とから格納されることを示している。

【００４６】

図８は、本発明の第１の実施の形態におけるソフトウェア開発装置の動作を、実行されるプログラムのメモリイメージとして説明する図である。図８（ａ）は、標準的な開発ツールを用いて作成されたロードモジュールに対応するメモリイメージを示している。

【００４７】

図８（ｂ）は、本実施の形態におけるソフトウェア開発装置によって作成されたロードモジュールに対応するメモリイメージを示している。命令抽出部１０２によって抽出された命令列（抽出命令列）と、初期化部とがリンクされている。抽出命令列は、初期化部に含まれるプログラムがＣＰＵコア２によって実行されることにより、図８（ｃ）に示すように半導体記憶装置１に設定される。なお、初期化部をロードモジュールにリンクしているが、ロードモジュールに相当する

機能をＯＳ（Operating System）のローダで実現することも可能である。

【００４８】

以上説明したように、本実施の形態における半導体記憶装置によれば、ＣＰＵコア２が実行する命令列の一部を命令バッファ１３に格納し、ＣＰＵコア２が命令バッファ１３に格納された命令列をフェッチするようにしたので、実行される頻度の高い命令列を命令バッファ１３に格納することによって、命令キャッシュのリプレースを減らして実行サイクル数を削減でき、処理性能を向上させることが可能となった。また、命令キャッシュのアクセス頻度を減らすことができるので、電力効率を高めることが可能となった。

【００４９】

また、半導体記憶装置１は、メモリマップドデバイスとしても動作するので、標準的なＣＰＵコアであってもリソースの読出し／書込み、動作モードの設定などを行なうことが可能である。

【００５０】

また、半導体記憶装置１に対するアクセスの可能性が低い場合に、半導体記憶装置１をバイパスし、ＣＰＵコア２と命令キャッシュ３とが直接接続されるように動作モードを設定することによって、半導体記憶装置１の消費電力を抑えることが可能となる。その結果、半導体記憶装置１を導入することによって発生する消費電力の増加を低減することができる。また、半導体記憶装置１が非アクティブ状態の場合においてのみ、初期化を含めた命令バッファ３の内容の書換えを可能としたので、リソースの書換えに伴う排他制御を、新たな機構を追加しなくても実現することが可能となった。

【００５１】

また、命令バッファ１３または命令キャッシュ３への命令アドレスの供給において、不要な方の命令アドレスの供給を停止するようにしたので、アドレスバスの変化によるメモリの不必要な状態遷移を抑制することができ、消費電力を削減することが可能となった。

【００５２】

また、本実施の形態におけるソフトウェア開発装置によれば、既存のソフトウ

ェア開発ツールに、半導体記憶装置１に配置される命令列を抽出する機能を追加するだけで、本実施の形態における半導体記憶装置１に対応したソフトウェアの開発ができるようになった。すなわち、既存のソフトウェア開発ツールの簡単な拡張のみで本実施の形態におけるソフトウェア開発装置を実現することが可能となった。

【００５３】

また、本実施の形態における半導体記憶装置１を用いたプロセッサシステムは、半導体記憶装置１に対するアクセスサイクルが保証されるので、ソフトウェア開発者が本実施の形態におけるソフトウェア開発装置を用いることによって、リアルタイム性の高いソフトウェアを開発することが可能になった。

【００５４】

さらには、本実施の形態におけるソフトウェア開発装置は、アセンブラおよびリンカを含む標準的なツールチェーンを活用し、幾つかのフィルタプログラムを追加するだけで実現できるので，ソフトウェア開発プログラムを容易に作成することが可能である。したがって、本実施の形態における半導体記憶装置１の機能を簡単に利用することが可能となった。

【００５５】

（第２の実施の形態）

本発明の第２の実施の形態におけるソフトウェア開発装置の構成は、図４に示す実施の形態１におけるソフトウェア開発装置の構成と同様である。したがって、重複する構成および機能の詳細な説明は繰返さない。

【００５６】

図９は、本発明の第２の実施の形態におけるソフトウェア開発装置の機能的構成を示すブロック図である。図５に示すソフトウェア開発装置と比較して、コンパイラ１０７およびコンパイラ１０７によって生成されたオブジェクトファイルをアセンブラソースの形式に変換して整形するオブジェクトファイル整形部１０８が追加されている点のみが異なる。したがって、重複する構成および機能の詳細な説明は繰返さない。

【００５７】

図１０は、本発明の第２の実施の形態におけるソフトウェア開発装置の処理手順を説明するためのフローチャートである。本実施の形態におけるソフトウェア開発装置の処理手順は、図６に示すフローチャートのステップＳ１１を、図１０に示すステップＳ２１～Ｓ２３に置換したものである。したがって、重複する処理手順の詳細な説明は繰返さない。なお、本実施の形態においては、Ｃ言語などの高級言語を用いたプログラム開発を想定している。

【００５８】

まず、プログラマが、オリジナルの高級言語のソースファイルの中から、キーボード２５、マウス２６などを用いて半導体記憶装置１に配置するべきプログラムの箇所を指定する（Ｓ２１）。

【００５９】

図１１は、Ｃ言語を用いたソフトウェア開発において、半導体記憶装置１に配置されるプログラムの一部を示す図である。半導体記憶装置１に配置される箇所は、“＃ｐｒａｇｍａ＿ｓｔａｓｈ１＿ｔｏｐ”と“＃ｐｒａｇｍａ＿ｓｔａｓｈ１＿ｅｎｄ”とで囲まれた部分である。ここで、“＿ｓｔａｓｈ１＿ｔｏｐ”と“＿ｓｔａｓｈ１＿ｅｎｄ”とは、予約された制御文字列を示している。

【００６０】

次に、コンパイラ１０７は、高級言語プログラムをコンパイルしてオブジェクトファイルを生成する（Ｓ２２）。このコンパイルは、高級言語を用いた通常のソフトウェア開発と同様であるので、詳細な説明は行なわない。

【００６１】

次に、オブジェクトファイル整形部１０８は、コンパイラ１０７によって生成されたオブジェクトファイルを、ソース行情報と共にディスアセンブルする。ディスアセンブルされた結果は、コンパイラ１０７が生成したアセンブリ言語レベルのプログラムに相当する情報が含まれているので、図７（ａ）に示すアセンブラソースの形式に変換することができる（Ｓ２３）。このようにして整形され、生成されたアセンブラソースに対して、図６のステップＳ１２以降の処理が行なわれる。

【００６２】

以上説明したように、本実施の形態におけるソフトウェア開発装置によれば、高級言語プログラムをコンパイルしてオブジェクトファイルを生成した後、オブジェクトファイルをディスアセンブルして整形するようにしたので、本発明の第１の実施の形態において説明した効果に加えて、高級言語を用いたプログラムであっても、半導体記憶装置１に用いたプロセッサシステムに対応したソフトウェア開発を行なうことが可能となった。

【００６３】

（第３の実施の形態）

本発明の第３の実施の形態における半導体記憶装置は、２組のプログラムを半導体記憶装置内に配置することが可能なものである。なお、３組以上のプログラムを配置する半導体記憶装置は、本実施の形態における半導体記憶装置から容易に推測可能であるため、その詳細な説明は行なわない。

【００６４】

図１２は、本発明の第３の実施の形態における半導体記憶装置の概略構成を示すブロック図である。この半導体記憶装置１ａは、アドレステーブル１（４１）と、アドレステーブル２（４２）と、判定器４３および４４と、命令バッファ１（４５）と、命令バッファ２（４６）と、セレクタ４７とを含む。

【００６５】

アドレステーブル１（４１）およびアドレステーブル２（４２）は、図２に示すアドレステーブル１１と同じ構成および機能を有する。また、判定器４３および４４は、図２に示す判定器１２と同じ構成および機能を有する。さらには、命令バッファ１（４５）および命令バッファ２（４６）は、命令バッファ１３と同じ構成および機能を有する。

【００６６】

セレクタ４７は、ＣＰＵコア２から出力された命令アドレスが、アドレステーブル１（４１）に格納されたアドレスにヒットする場合、命令バッファ１（４５）から出力された命令コードを選択して、ＣＰＵコア２へ出力する。また、セレクタ４７は、ＣＰＵコア２から出力された命令アドレスが、アドレステーブル２（４２）に格納されたアドレスにヒットする場合、命令バッファ２（４６）から

出力された命令コードを選択して、ＣＰＵコア２へ出力する。さらには、セレクタ４７は、ＣＰＵコア２から出力された命令アドレスが、アドレステーブル１（４１）に格納されたアドレスにも、アドレステーブル２（４２）に格納されたアドレスにもヒットしない場合、命令キャッシュ３から出力された命令コードを選択して、ＣＰＵコア２へ出力する。

図１３は、ＣＰＵコア２が半導体記憶装置１ａをメモリマップドデバイスとしてアクセスする場合のメモリマップを示す図である。図３に示すメモリマップと比較して、“アドレステーブル”および“命令バッファ”がそれぞれ２つずつマッピングされており、各々４Ｋバイトの領域となっている点のみが異なる。

【００６７】

図１４は、本発明の第３の実施の形態におけるソフトウェア開発装置の処理手順を説明するためのフローチャートである。まず、ディスアセンブラが、本実施の形態における半導体記憶装置１ａを含むプロセッサシステムが実行するソフトウェアのオブジェクトファイル（ａ．ｏ）５１をディスアセンブルする（Ｓ３１）。

【００６８】

次に、ディスアセンブラによって生成されたソースファイルの中から複数箇所で発生する共通命令列が検出される（Ｓ３２）。検出された共通命令列および共通命令列に関連するアドレス情報はそれぞれアセンブルされて、オブジェクトファイル（ｂｕｆ．ｏ，ａｔ．ｏ）５３および５５が生成される。

【００６９】

最後に、生成されたオブジェクトファイル（ｂｕｆ．ｏ，ａｔ．ｏ）５３および５５は、オリジナルのオブジェクトファイル（ａ．ｏ）５１や、半導体記憶装置１の初期化のためのルーチン（ｓｔａｓｈ．ｏ）５４と共にリンクされて、ロードモジュール（ａ．ａｂｓ）５６が生成される。なお、ディスアセンブラおよびリンカは、ターゲットとなるＣＰＵコア２のソフトウェア開発用ツールがそのまま利用可能である。

【００７０】

また、共通命令列の検出においては、ターゲットとなる半導体記憶装置１のサイズ情報５２などのハードウェア構成の情報をパラメータとして用い、繰返し発生する命令パターンの中から、半導体記憶装置１に格納できるサイズの共通命令列が抽出される。抽出された共通命令列は、命令バッファ１（４５）および命令バッファ２（４６）のそれぞれに配置される。

【００７１】

以上説明したように、本実施の形態における半導体記憶装置１によれば、半導体記憶装置１内に複数の共通命令列を配置できるので、同じ動作を行なう共通プログラム部分について第１の実施の形態において説明した効果が得られ、さらに半導体記憶装置１の汎用性を高めることが可能となった。

【００７２】

（第４の実施の形態）

本発明の第４の実施の形態における半導体記憶装置は、図２に示す第１の実施の形態における半導体記憶装置と比較して、判定器の構成および機能のみが異なる。したがって、重複する構成および機能の詳細な説明は繰返さない。なお、本実施の形態における判定器の参照符号を１２’として説明する。

【００７３】

図１５は、本発明の第４の実施の形態における判定器１２’の構成を示すブロック図である。この判定器１２’は、比較器６１と、減算器６２および６３と、減算器６２による減算結果が格納される差レジスタ１（６４）と、減算器６３による減算結果が格納される差レジスタ２（６５）と、インバータ６６と、２入力ＡＮＤ回路６７とを含む。

【００７４】

比較器６１は、有効ビット１１１が設定されている場合、ＣＰＵコア２が出力したアドレスの上位１９ビットと、アドレスの最大値１１２の上位１９ビットとを比較する。アドレスの上位１９ビットが一致する場合、比較器６１は減算器６２および６３をイネーブルにする。また、アドレスの上位１９ビットが一致しない場合には、比較器６１は減算器６２および６３をディスエーブルにする。

【００７５】

減算器６２は、ＣＰＵコア２が出力したアドレスの下位１３ビットからアドレスの最大値１１２の下位１３ビットを減算し、その減算結果を差レジスタ１（６４）に格納する。また、減算器６３は、ＣＰＵコア２が出力したアドレスの下位１３ビットからアドレスの最小値１１３の下位１３ビットを減算し、その減算結果を差レジスタ２（６５）に格納する。

【００７６】

ＡＮＤ回路６７は、差レジスタ１（６４）の符号ビットと、差レジスタ２（６５）の符号ビットを反転した値との論理和を演算することにより、ＣＰＵコア２が出力したアドレスの下位１３ビットが所定の範囲に含まれているか否かを判定する。命令アドレスの下位１３ビットがヒットした場合には、ＡＮＤ回路６７はＨレベルを出力する。

【００７７】

以上説明したように、本実施の形態における半導体記憶装置１によれば、比較器６１がＣＰＵコア２が出力したアドレスの上位１９ビットと、アドレスの最大値１１２の上位１９ビットとを比較し、比較結果に応じて減算器６２および６３を制御するようにしたので、減算器６２および６３と、差レジスタ１（６４）と、差レジスタ２（６５）とのビット数を削減することができ、判定器１２’の回路規模を削減することが可能になると共に、半導体記憶装置１の消費電力を削減することが可能となった。

【００７８】

（第５の実施の形態）

本発明の第５の実施の形態における半導体記憶装置は、図２に示す第１の実施の形態における半導体記憶装置と比較して、判定器の構成および機能のみが異なる。したがって、重複する構成および機能の詳細な説明は繰返さない。なお、本実施の形態における判定器の参照符号を１２”として説明する。

【００７９】

図１６は、本発明の第５の実施の形態における判定器１２”の構成を示すブロック図である。この判定器１２”は、減算器７１および７２と、減算器７１による減算結果が格納される差レジスタ１（７３）と、減算器７２による減算結果が

格納される差レジスタ２（７４）と、インバータ７５と、３入力ＡＮＤ回路７６とを含む。

【００８０】

本実施の形態においては、命令バッファ１３に配置されるプログラムを基本ブロックに限定、すなわち命令バッファ１３に配置される命令列に分岐命令が含まれず、命令バッファ１３に配置される命令列への分岐も起こらないように限定を加えるものである。

【００８１】

ソフトウェア開発者に対して、このような限定を加えることにより、命令バッファ１３に対する命令コードのフェッチは必ず命令アドレスがインクリメントされることになる。すなわち、一旦命令アドレスがヒットすれば、次にミスヒットするまで差レジスタ２（７４）のＭＳＢの値は変化しない。そこで、次にミスヒットが発生するまで、ＣＰＵコア２が出力する命令アドレスからアドレスの最小値１１３を減算する減算器７２の動作を停止させるものである。

【００８２】

ＡＮＤ回路７６は、差レジスタ１（７３）の符号ビットと、差レジスタ２（７４）の符号ビットを反転した値との論理和を演算することにより、ＣＰＵコア２が出力したアドレスが所定の範囲に含まれているか否かを判定する。命令アドレスがヒットした場合には、ＡＮＤ回路７６はＨレベルを出力して減算器７２をディスエーブルにして動作を停止させる。また、命令アドレスがミスヒットした場合には、ＡＮＤ回路７６がＬレベルを出力して、減算器７２をイネーブルにして動作を開始させる。

【００８３】

また、命令バッファ１３には、ＣＰＵコア２が出力する命令アドレスの下位１２ビットが供給される。

【００８４】

図１７は、本発明の第５の実施の形態におけるソフトウェア開発装置の処理手順を説明するためのフローチャートである。本実施の形態におけるソフトウェア開発装置の処理手順は、図６に示すフローチャートのステップＳ１５を、図１７

に示すステップＳ３１～Ｓ３３に置換したものである。したがって、重複する処理手順の詳細な説明は繰返さない。

【００８５】

アドレス情報取得部１０３は、ステップＳ１４において取得したアドレス情報から、先頭命令のアドレスの下位１２ビットを取得する（Ｓ３１）。そして、命令列抽出部１０２によって抽出された命令列（プログラム）を、先頭命令のアドレスの下位１２ビットに相当するバイト数だけ命令バッファ１３内においてずらす。このようにして、ＣＰＵコア２から出力される命令アドレスの下位１２ビットと、命令バッファ１３のアドレスの下位１２ビットとの整合をとるようにしている。

【００８６】

次に、アセンブラ１０５は、命令列抽出部１０２によって抽出された命令列をアセンブルし（Ｓ３３）、図６のステップＳ１６の処理へ進む。

【００８７】

以上説明したように、ソフトウェア開発者にプログラム作成時に制限を課すことによって、判定器１２”における判定を容易に行なうことができると共に、命令アドレスがヒットしている間は減算器７２の動作を停止させることができるので、半導体記憶装置１の消費電力を削減することが可能となった。

【００８８】

（第６の実施の形態）

ソフトウェアのデバッグで広く使用されているブレークポイント機能は、通常、ブレークポイントに相当する命令を例外命令に置換することによって実現される。しかし、本発明の第１の実施の形態における半導体記憶装置１においては、プロセッサシステムが動作中に命令バッファ１３などの書換えが不可能であり、命令バッファ１３に例外命令を設定することができない。したがって、プログラムのデバッグが困難になるといった問題点がある。本実施の形態においては、この問題点を解消してプログラムのデバッグを効率良く行なえるようにしたものである。

【００８９】

本発明の第６の実施の形態における半導体記憶装置を用いたプロセッサシステムの構成は、第５の実施の形態における半導体記憶装置を用いたプロセッサシステムの構成と比較して、半導体記憶装置用予約領域がメモリマップにマッピングされている点のみが異なる。したがって、重複する構成および機能の詳細な説明は繰返さない。

【００９０】

図１８は、本発明の第６の実施の形態におけるメモリマップの一例を示す図である。主記憶５に、半導体記憶装置１の命令バッファ１３と同じサイズの半導体記憶装置用予約領域が設定される。

【００９１】

本発明の第６の実施の形態におけるソフトウェア開発装置の処理手順は、第５の実施の形態におけるソフトウェア開発装置の処理手順と比較して、ステップＳ１８の処理を除いて同じである。したがって、重複する処理手順の詳細な説明は繰返さない。なお、本実施の形態において、ステップＳ１８をステップＳ１８’として説明する。

【００９２】

ステップＳ１８’において、リンカ１０６は、命令バッファ１３に配置される命令列に対応する主記憶５の領域の先頭アドレスに分岐命令を設定する。その分岐命令の分岐先は、図１８示す半導体記憶装置用予約領域の先頭アドレスとする。また、命令列抽出部１０２によって抽出された命令列（抽出命令列）の末尾に分岐命令を追加して半導体記憶装置用予約領域に設定する。その分岐命令の分岐先は、抽出命令列に対応する主記憶領域の最後尾アドレスの直後に設定する。

【００９３】

この処理を，図７に示すプログラムを用いて具体的に説明する。まず、ステップＳ１８’の前半の処理によって、０ｘ２６４番地に０ｘｆｆ０００への分岐命令が設定される。そして、ステップＳ１８’の後半の処理によって、図７（ｂ）に示す抽出命令列の直後に０ｘ２６ｃ番地への分岐命令が追加され、この命令列が０ｘｆｆ０００番地に設定される。

【００９４】

このようにして、デバッグ用の初期化部をリンクした場合、分岐命令以外の抽出命令列本体の実行は、主記憶５の半導体記憶装置用予約領域からの命令フェッチによって行なわれる。この領域は主記憶５上にあるので、ブレークポイントの設定が可能である。すなわち、プログラム本体と抽出命令列はそのままで、初期化部だけ異なるプログラムを利用すれば、プログラム本体と抽出命令列のアルゴリズムをそのまま用いて、ソフトウェアのデバッグが行なえるようになる。

【００９５】

以上説明したように、本実施の形態におけるプロセッサシステムによれば、半導体記憶装置用予約領域を主記憶５上に設け、抽出命令列を半導体記憶装置用予約領域に配置するようにしたので、ブレークポイントを設定することができ、ソフトウェアのデバッグを容易に行なうことが可能となった。

【００９６】

今回開示された実施の形態は、すべての点で例示であって制限的なものではないと考えられるべきである。本発明の範囲は上記した説明ではなくて特許請求の範囲によって示され、特許請求の範囲と均等の意味および範囲内でのすべての変更が含まれることが意図される。

【００９７】

【発明の効果】

本発明の半導体記憶装置によれば、セレクタが、判定器の判定結果に応じて、命令バッファに格納される命令コードと命令キャッシュに格納される命令コードとを選択的に出力するので、プロセッサが命令バッファに格納された命令をフェッチする場合にはアクセスサイクルが保証されることでリアルタイム性の高いソフトウェアの開発が容易となると共に、命令キャッシュの動作が行なわれないので電力効率を向上させることが可能となった。

【００９８】

本発明のソフトウェア開発装置によれば、抽出手段が、ソースファイルの中から、半導体記憶装置に配置する命令列を抽出する。そして、生成手段が、ソースファイル、抽出手段によって抽出された命令列および整形手段によって整形されたアドレス情報からロードモジュールを生成するので、プログラマは半導体記憶

装置を含んだプロセッサシステムのソフトウェア開発を容易に行なうことが可能となった。

【図面の簡単な説明】

【図１】　本発明の第１の実施の形態における半導体記憶装置が利用されるプロセッサシステムの構成例を示すブロック図である。

【図２】　本発明の第１の実施の形態における半導体記憶装置１の内部構成を示すブロック図である。

【図３】　ＣＰＵコア２が半導体記憶装置１をメモリマップドデバイスとしてアクセスする場合のメモリマップを示す図である。

【図４】　本発明の第１の実施の形態における半導体記憶装置１を用いたプロセッサシステムのソフトウェア開発装置の構成を示すブロック図である。

【図５】　本発明の第１の実施の形態におけるソフトウェア開発装置の機能的構成を示すブロック図である。

【図６】　本発明の第１の実施の形態におけるソフトウェア開発装置の処理手順を説明するためのフローチャートである。

【図７】　（ａ）は、ＣＰＵコア２によって実行されるプログラムの一部を示す図である。（ｂ）は、命令列抽出部１０２によって抽出されたプログラムの一例を示す図である。（ｃ）は、アドレス情報整形部１０４によって整形されたアドレス情報を示す図である。（ｄ）は、リンカ１０６によってリンクされたモジュールのディスアセンブル結果の一部を示す図である。

【図８】　本発明の第１の実施の形態におけるソフトウェア開発装置の動作を、実行されるプログラムのメモリイメージとして説明する図である。

【図９】　本発明の第２の実施の形態におけるソフトウェア開発装置の機能的構成を示すブロック図である。

【図１０】　本発明の第２の実施の形態におけるソフトウェア開発装置の処理手順を説明するためのフローチャートである。

【図１１】　Ｃ言語を用いたソフトウェア開発において、半導体記憶装置１に配置されるプログラムの一部を示す図である。

【図１２】　本発明の第３の実施の形態における半導体記憶装置の概略構成

を示すブロック図である。

【図１３】　ＣＰＵコア２が半導体記憶装置１ａをメモリマップドデバイスとしてアクセスする場合のメモリマップを示す図である。

【図１４】　本発明の第３の実施の形態におけるソフトウェア開発装置の処理手順を説明するためのフローチャートである。

【図１５】　本発明の第４の実施の形態における判定器１２’の構成を示すブロック図である。

【図１６】　本発明の第５の実施の形態における判定器１２”の構成を示すブロック図である。

【図１７】　本発明の第５の実施の形態におけるソフトウェア開発装置の処理手順を説明するためのフローチャートである。

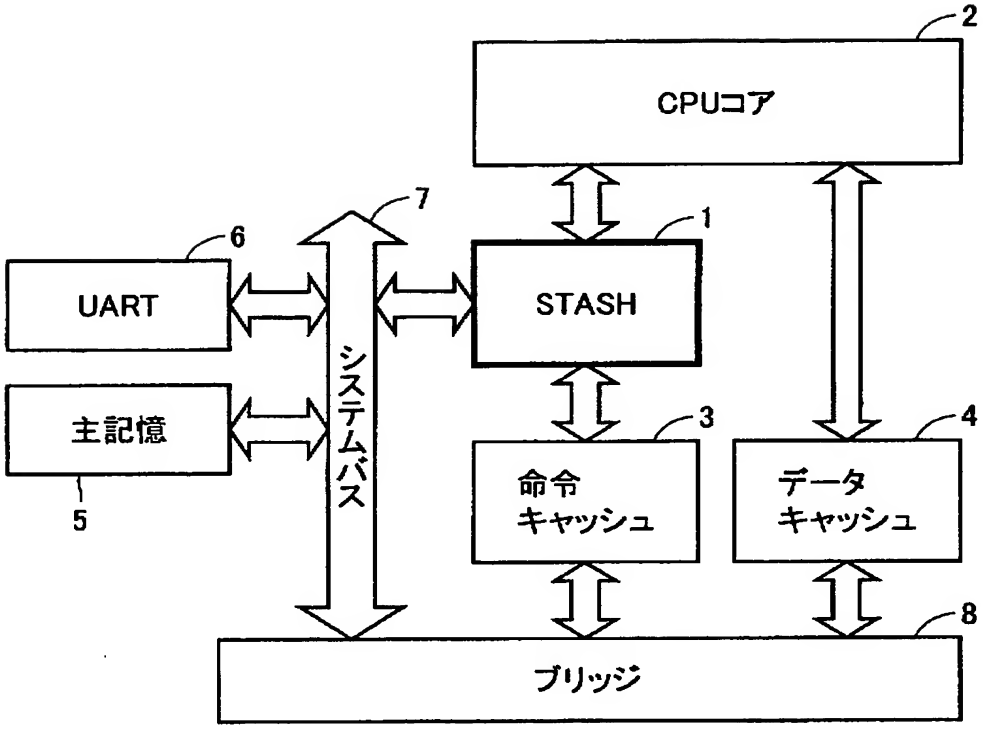
【図１８】　本発明の第６の実施の形態におけるメモリマップの一例を示す図である。

【符号の説明】

１　半導体記憶装置、２　ＣＰＵコア、３　命令キャッシュ、４　データキャッシュ、５　主記憶、６　ＵＳＡＲＴ、７　システムバス、８　ブリッジ、１１，４１，４２　アドレステーブル、１２，１２’，１２”，４３，４４　判定器、１３，４５，４６　命令バッファ、１４，４７　セレクタ、１５，１６　ＳＷ、１７，６６，７５，１２５　インバータ、２１　コンピュータ本体、２２　ディスプレイ装置、２３　ＦＤドライブ、２４　ＦＤ、２５　キーボード、２６　マウス、２７　ＣＤ－ＲＯＭ装置、２８　ＣＤ－ＲＯＭ、２９　ネットワーク通信装置、５１　オブジェクト、５２　サイズ情報、５３　共通命令列、５４　初期化部、５５　アドレス情報、５６　ロードモジュール、６１　比較器、６２，６３，７１，７２，１２１，１２２　減算器、６４，６５，７３，７４，１２３，１２４　差レジスタ、６７，７６，１２６　ＡＮＤ、１０１　ソースファイル格納部、１０２　命令列抽出部、１０３　アドレス情報取得部、１０４　アドレス情報整形部、１０５　アセンブラ、１０６　リンカ、１０７　コンパイラ、１０８　オブジェクトファイル整形部、１１１　有効ビット、１１２　最大値、１１３　最小値。

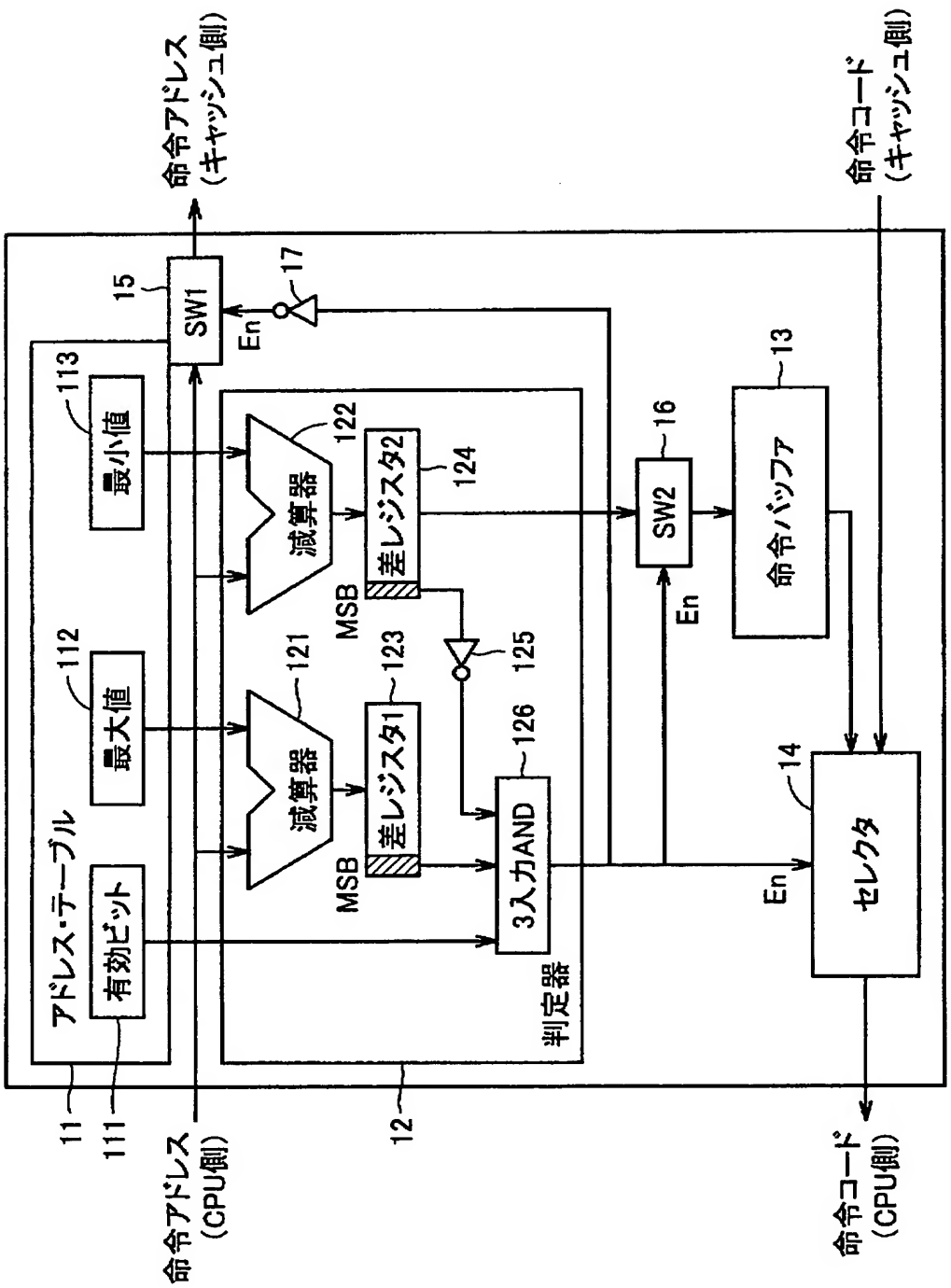
【書類名】　　図面

【図１】

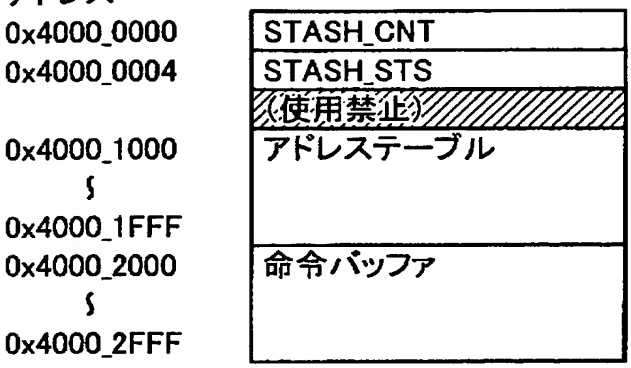


出証特２００３－３００５６４９

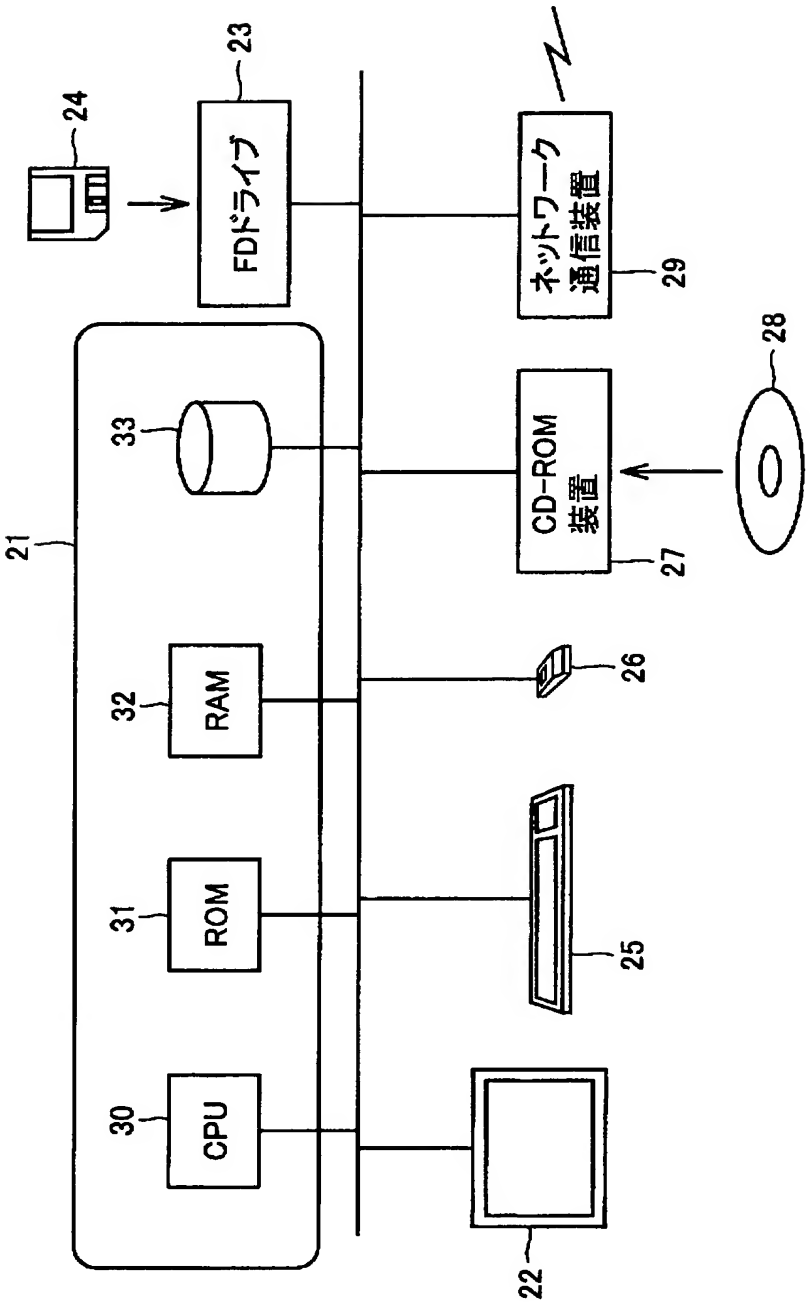
【図２】


命令アドレス
(キャッシュ側)
命令コード
(キャッシュ側)
15
SW1
17
En
113
最小値
112
最大値
アドレス・テーブル
有効ビット
121
減算器
122
減算器
MSB
MSB
123
差レジスタ1
124
差レジスタ2
125
126
3入力AND
判定器
16
SW2
En
13
命令バッファ
14
セレクタ
En
11
111
12
命令アドレス
(CPU側)
命令コード
(CPU側)

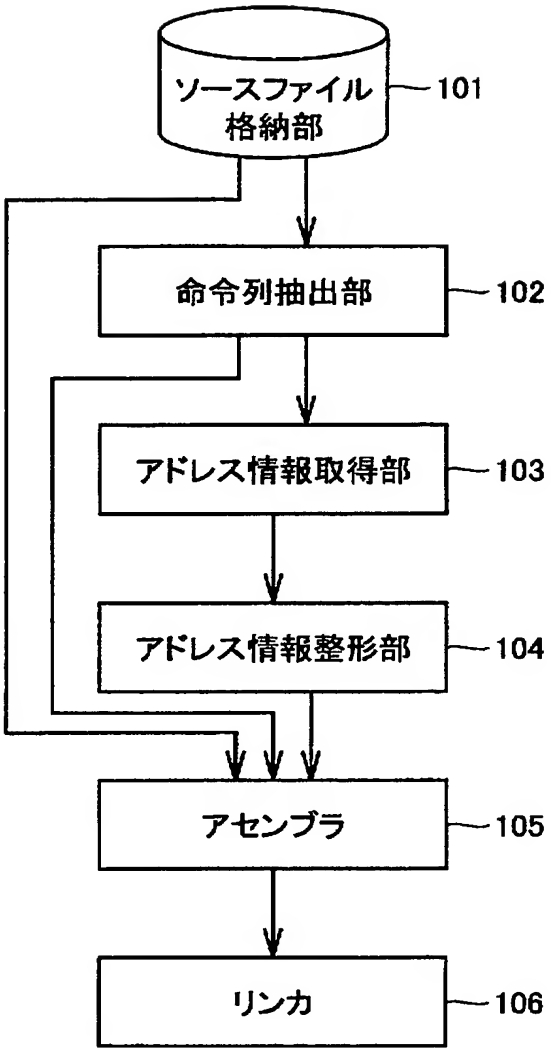
【図３】


アドレス
0x4000_0000
0x4000_0004
0x4000_1000
0x4000_1FFF
0x4000_2000
0x4000_2FFF
STASH_CNT
STASH_STS
(使用禁止)
アドレステーブル
命令バッファ

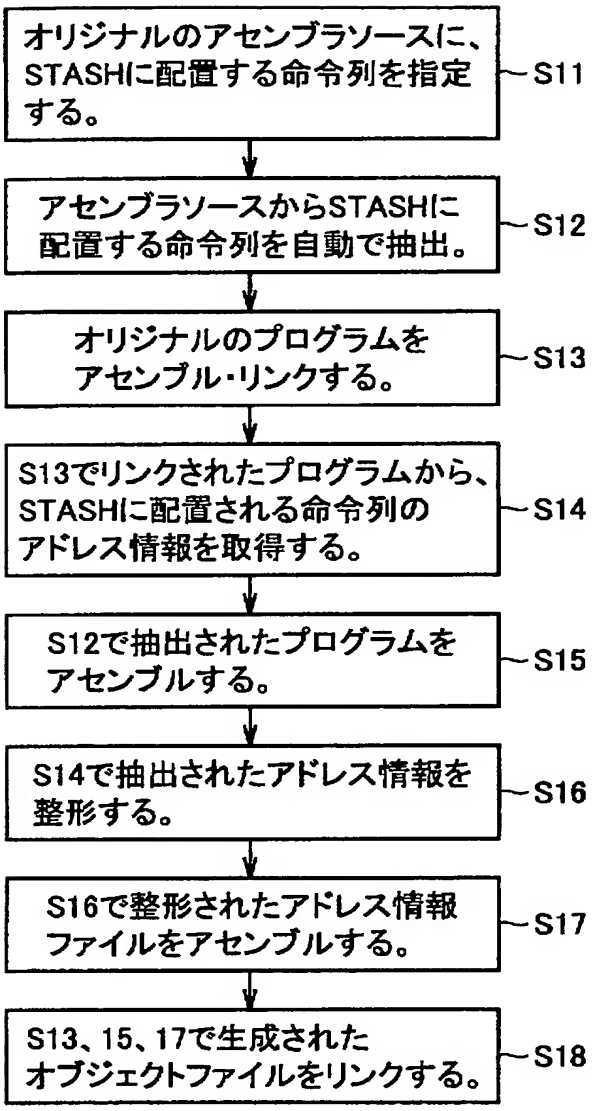
【図４】


23
24
FDドライブ
ネットワーク
通信装置
29
28
33
CD-ROM
装置
27
21
26
32
RAM
31
ROM
25
30
CPU
22

【図５】


ソースファイル
格納部
101
命令列抽出部
102
アドレス情報取得部
103
アドレス情報整形部
104
アセンブラ
105
リンカ
106

【図６】


オリジナルのアセンブラソースに、STASHに配置する命令列を指定する。
S11
アセンブラソースからSTASHに配置する命令列を自動で抽出。
S12
オリジナルのプログラムをアセンブル・リンクする。
S13
S13でリンクされたプログラムから、STASHに配置される命令列のアドレス情報を取得する。
S14
S12で抽出されたプログラムをアセンブルする。
S15
S14で抽出されたアドレス情報を整形する。
S16
S16で整形されたアドレス情報ファイルをアセンブルする。
S17
S13、15、17で生成されたオブジェクトファイルをリンクする。
S18

【図７】

(a)

```
.text
_func1:
_stash1_top:
 ld  r1,  @r14+
 ld  r2,  @r14+
_stash1_end:
 add  r1,  r2
 （以下略）
```

(b)

```
 ld  r1,  @r14+
 ld  r2,  @r14+
```

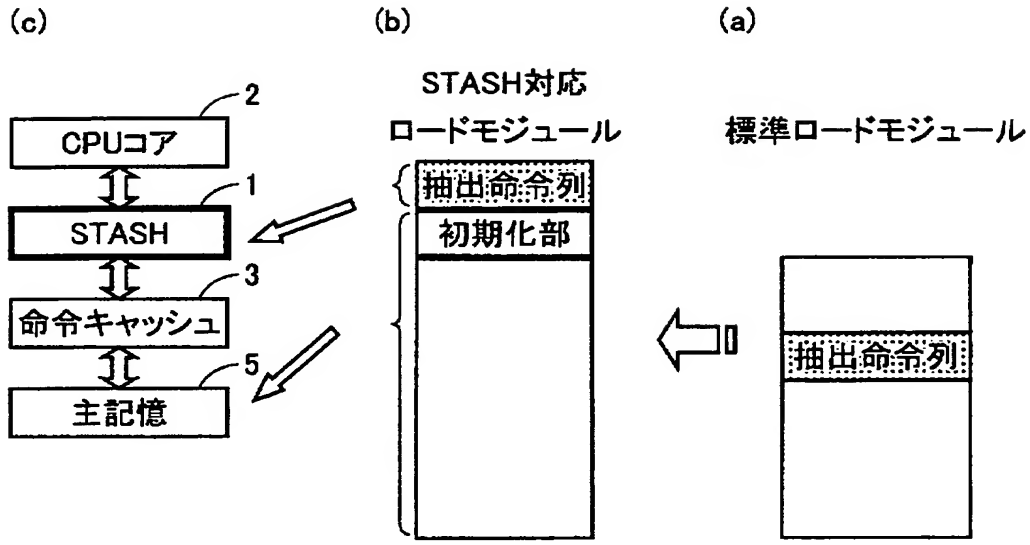
(c)

```
.section _stash_at,  "a"
.word 0x00000264, 0x0000026c
```

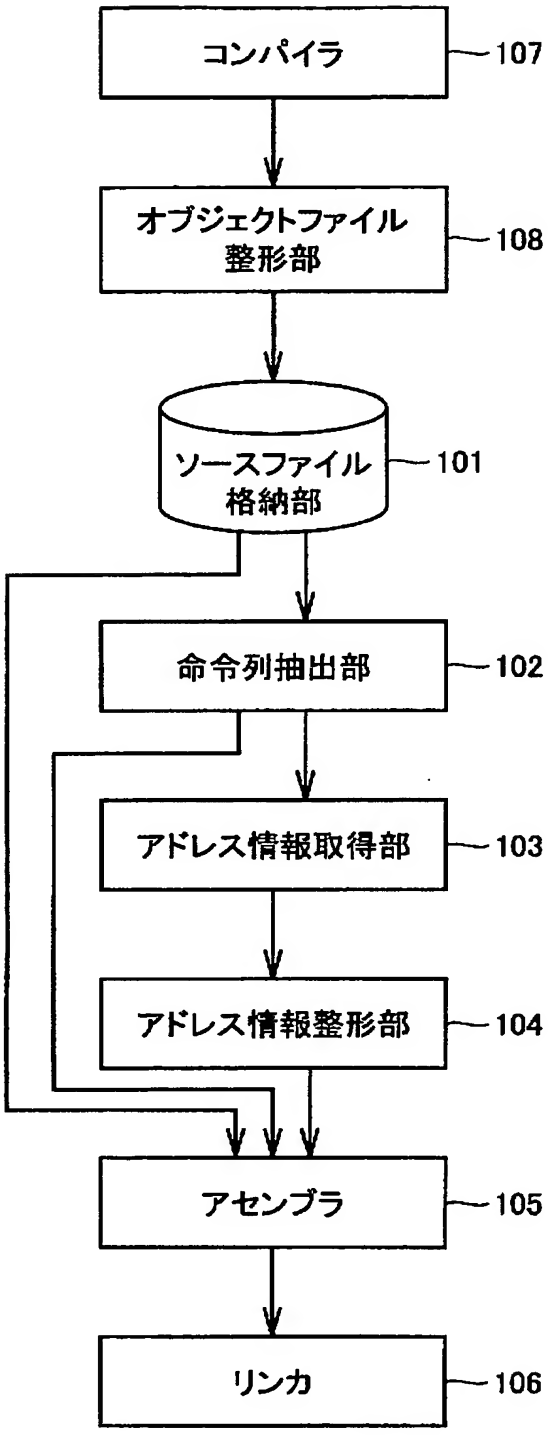
(d)

```
Disassembly of section  .stash_at:

40001000  <.stash_at>:
     0:  00  00  02  64        subv  r0,r0→  cmpeq  r2,r4
     4:  00  00  02  6c        subv  r0,r0→  cmpeq  r2,r12
Disassembly of section  .stash_buf:

40002000  <.stash_buf>:
     0:  21  ee  22  ee        ld  r1,@lr+→  ld  r2,@lr+
```

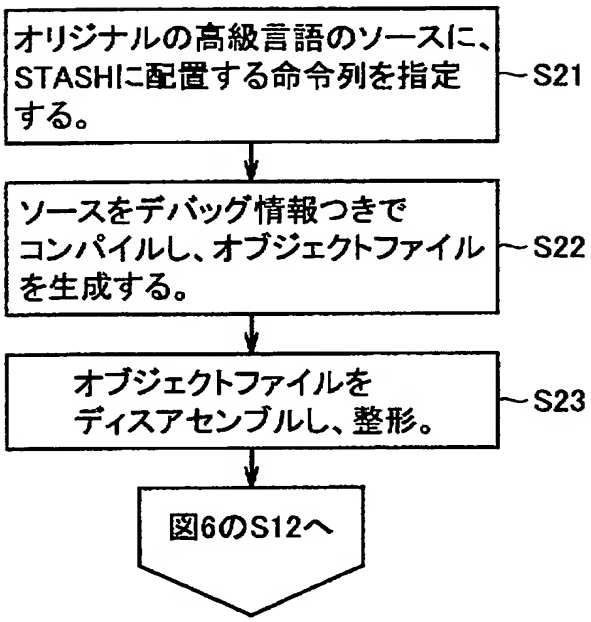
【図８】


(c)
(b)
(a)
STASH対応
ロードモジュール
標準ロードモジュール
2
CPUコア
1
STASH
3
命令キャッシュ
5
主記憶
抽出命令列
初期化部
抽出命令列

【図９】


コンパイラ
107
オブジェクトファイル
整形部
108
ソースファイル
格納部
101
命令列抽出部
102
アドレス情報取得部
103
アドレス情報整形部
104
アセンブラ
105
リンカ
106

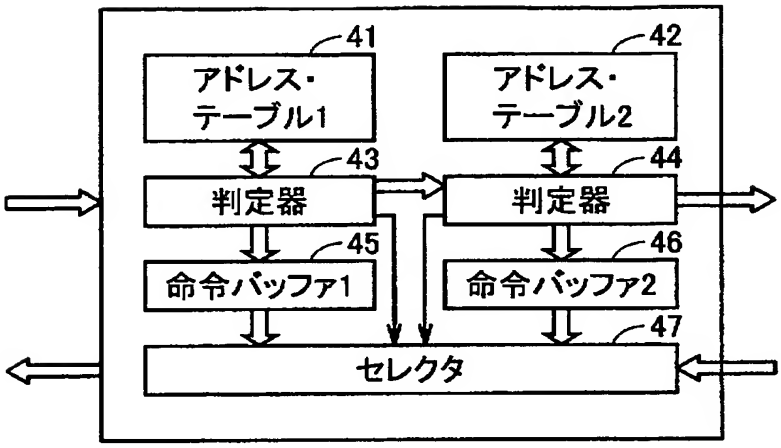
【図１０】



【図１１】

```
  for(i= 0;; i++){
#pragma _stash1_top          MACRO_A(i, j);
        if( j > INT_MAX)
            break;
    }
"#pragma_stash1_end
    return( j );
```

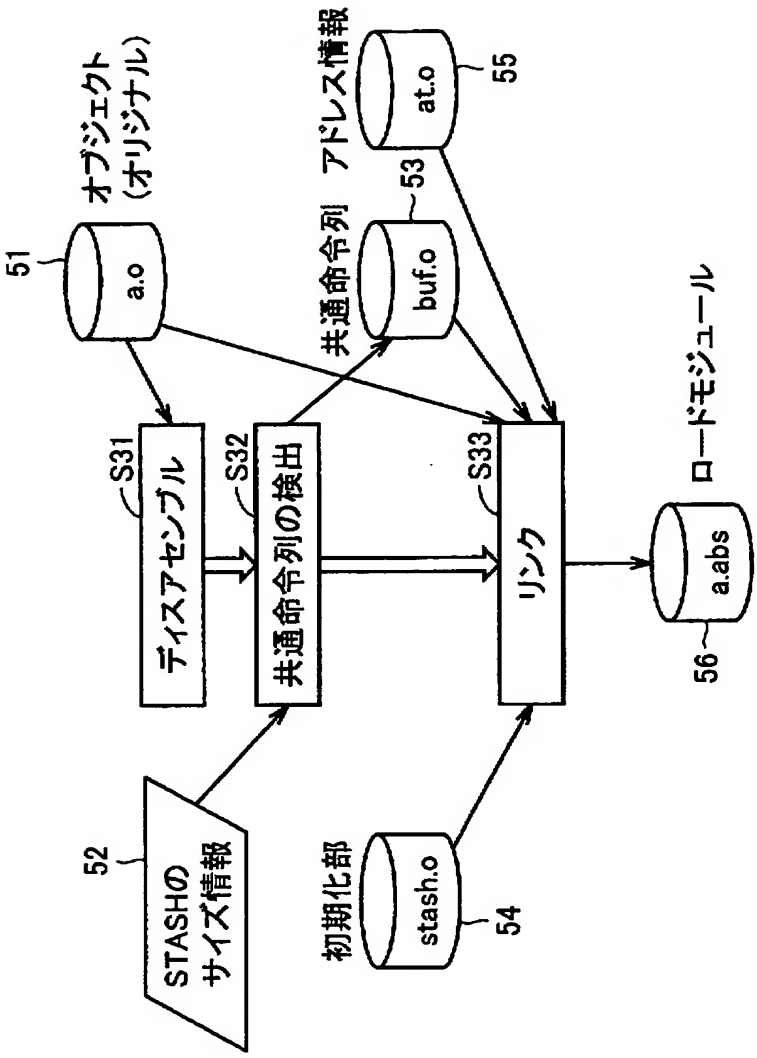
【図１２】



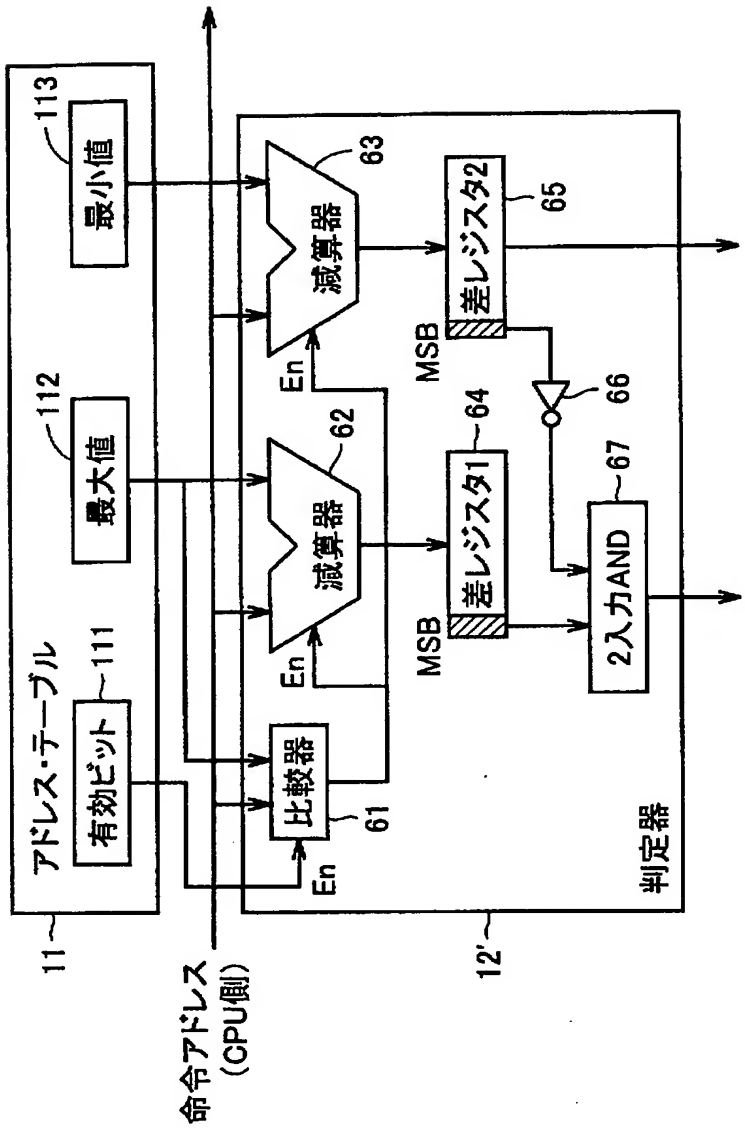
【図１３】

アドレス	
0x4000_0000	STASH_CNT
0x4000_0004	STASH_STS
	(使用禁止)
0x4000_1000	アドレステーブル1
｛	
0x4000_1FFF	
0x4000_2000	アドレステーブル2
｛	
0x4000_2FFF	
0x4000_3000	命令バッファ1
｛	
0x4000_3FFF	
0x4000_4000	命令バッファ2
｛	
0x4000_4FFF	

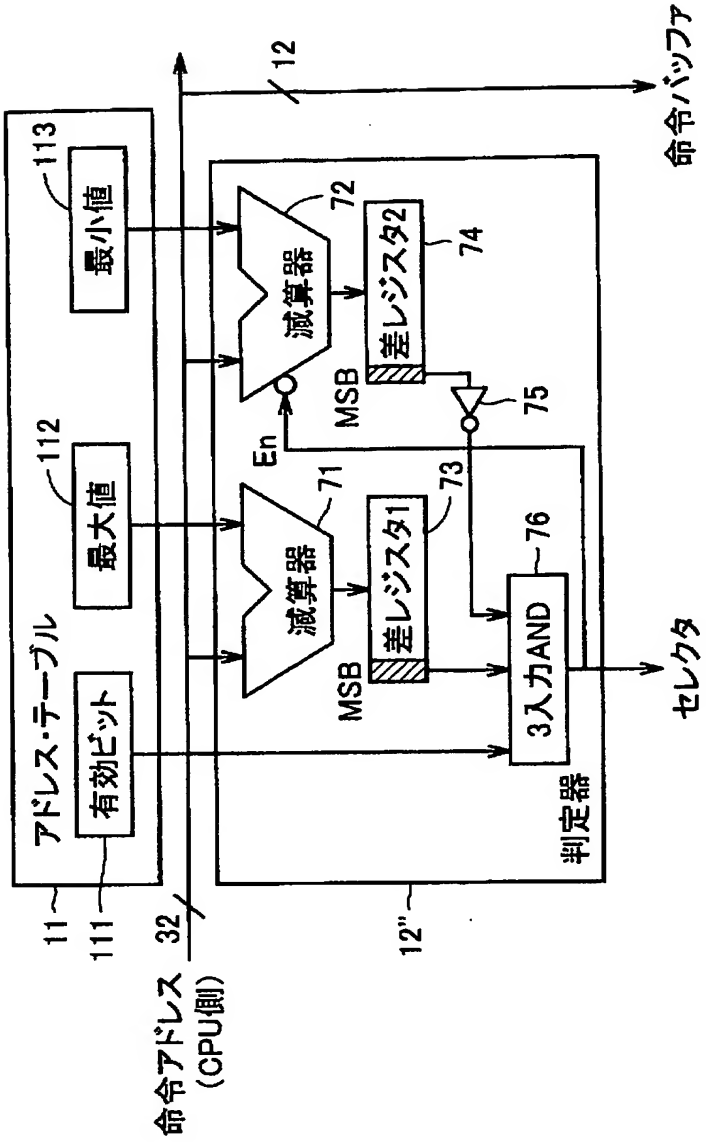
【図１４】


オブジェクト
(オリジナル)
51
a.o
アドレス情報
55
at.o
53
共通命令列
buf.o
ロードモジュール
S31
ディスアセンブル
S32
共通命令列の検出
S33
リンク
a.abs
56
52
STASHの
サイズ情報
初期化部
stash.o
54

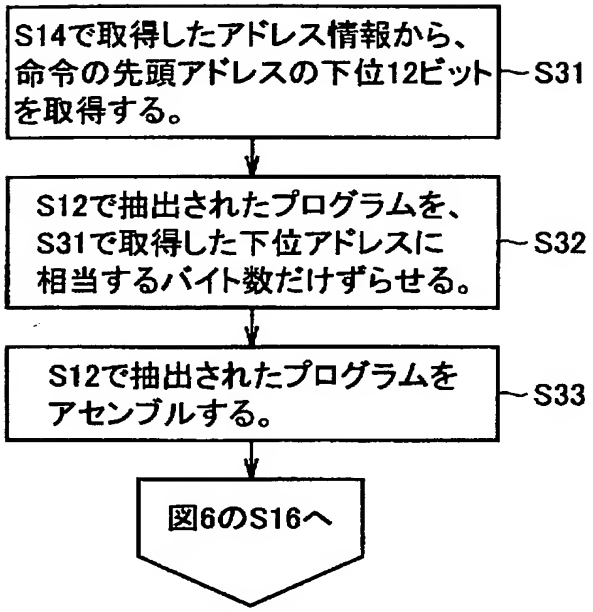
【図１５】


アドレス・テーブル
11
113
最小値
112
最大値
111
有効ビット
命令アドレス
(CPU側)
判定器
12'
63
減算器
62
減算器
61
比較器
En
En
En
MSB
MSB
64
差レジスタ1
65
差レジスタ2
66
67
2入力AND

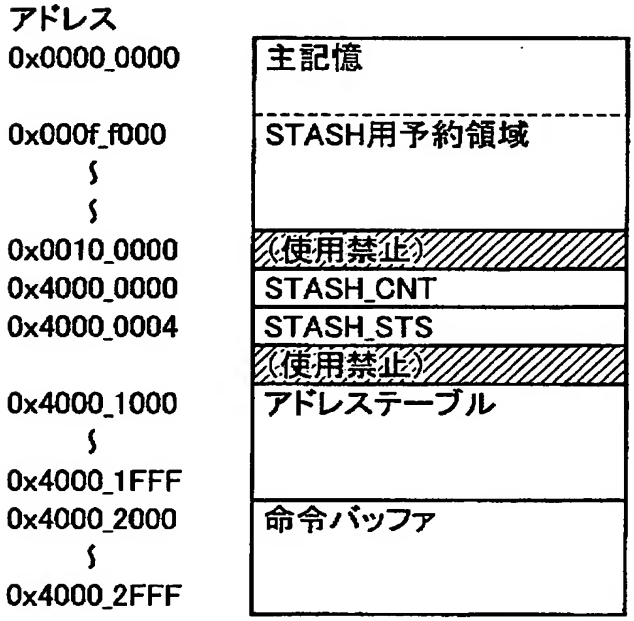
【図１６】


アドレス・テーブル
113
最小値
112
最大値
有効ビット
111
11
32
命令アドレス
(CPU側)
12
命令バッファ
12"
判定器
減算器
72
差レジスタ2
74
MSB
75
En
減算器
71
差レジスタ1
73
MSB
76
3入力AND
セレクタ

【図１７】


S14で取得したアドレス情報から、命令の先頭アドレスの下位12ビットを取得する。
S31
S12で抽出されたプログラムを、S31で取得した下位アドレスに相当するバイト数だけずらせる。
S32
S12で抽出されたプログラムをアセンブルする。
S33
図6のS16へ

【図１８】


アドレス
0x0000_0000
主記憶
0x000f_f000
STASH用予約領域
0x0010_0000
(使用禁止)
0x4000_0000
STASH_CNT
0x4000_0004
STASH_STS
(使用禁止)
0x4000_1000
アドレステーブル
0x4000_1FFF
0x4000_2000
命令バッファ
0x4000_2FFF

【書類名】　　要約書

【要約】

【課題】　命令フェッチ時におけるアクセスサイクルを保証でき、プロセッサシステムの電力効率を向上させることが可能な半導体記憶装置を提供すること。

【解決手段】　アドレステーブル１１に、命令バッファ１３に格納された命令列のアドレス範囲が設定される。判定器１２は、ＣＰＵコアから出力された命令アドレスがアドレステーブル１１に設定されたアドレス範囲内であるか否かを判定する。そして、セレクタ１４が、判定器１２の判定結果に応じて、命令バッファ１３に格納される命令コードと命令キャッシュに格納される命令コードとを選択的に出力する。したがって、ＣＰＵコアが命令バッファ１３に格納された命令をフェッチする場合にはアクセスサイクルが保証されると共に、命令キャッシュの動作が行なわれないので電力効率を向上させることが可能となる。

【選択図】　　図２

出　願　人　履　歴　情　報

識別番号　　〔０００００６０１３〕

１．変更年月日　　１９９０年　８月２４日
［変更理由］　新規登録
住　所　東京都千代田区丸の内２丁目２番３号
氏　名　三菱電機株式会社